Guard

Smart contracts security assessment

Final report

Tariff: Standard

LiquidLocker

April 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for LiquidNFTs team.

The audited code is available at @wise-foundation/liquidnfts-audit-scope/contracts Github repository and was audited after commit 6fc41c8. A recheck has been done after commit bb432ec.

The audited contract makes it possible to borrow tokens secured by NFT, as well as lend tokens at interest.

Late payment penalties are charged. If the borrower has missed payments by 7 days his NFT will transfer to either the single provider address or the trusted multisig to be auctioned.

Name	LiquidLocker
Audit date	2022-03-31 - 2022-04-02
Language	Solidity
Platform	Ethereum

Contracts checked

Name	Address
LiquidBase	
LiquidHelper	
LiquidLocker	
LiquidTransfer	

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyse smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed

Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

No issues were found

Medium severity issues

No issues were found

Low severity issues

1. Max rate is not checked (FIXED) (LiquidLocker)

The locker owner can use `updateSettings()` or `increasePaymentRate()` to set `globals.paymentRate` to be greater than the `RATE_MAX`.

```
    function updateSettings(
        uint256 _newPaymntRate,
        uint256 _newPaymentTime
    )
        external
        onlyLockerOwner
        onlyDuringContributionPhase
    {
        require(
            _newPaymntRate > globals.paymentRate,
            "LiquidLocker: INVALID_RATE"
        );

        globals.paymentRate = _newPaymntRate;

        ...
    }
```

```
    function increasePaymentRate(
        uint256 _newPaymntRate
    )
```

```
        external
        onlyLockerOwner
        onlyDuringContributionPhase
    {
        require(
            _newPaymntRate > globals.paymentRate,
            "LiquidLocker: INVALID_INCREASE"
        );

        globals.paymentRate = _newPaymntRate;

        ...
    }
```

Recommendation: We recommend checking for an upper limit for the `paymentRate` variable.

Update: The LiquidNFTs team addressed this issue in the following pull request: https://github.com/wise-foundation/liquidnfts-audit-scope/pull/1/files.

Conclusion

LiquidLocker LiquidBase, LiquidHelper, LiquidLocker, LiquidTransfer contracts were audited. 1 low severity issue was found.

The code is well documented and well written.

Update: The low severity issue was fixed in the update (pull request https://github.com/wise-foundation/liquidnfts-audit-scope/pull/1/files).

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability)set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Slither output

```
LiquidLocker.enableLocker(uint256) (LiquidLocker.sol#245-299) uses a dangerous strict
equality:
        - require(bool,string)(paymentTimeNotSet() == true,LiquidLocker:
ENABLED_LOCKER) (LiquidLocker.sol#256-259)
LiquidHelper.floorNotReached() (LiquidHelper.sol#45-51) uses a dangerous strict
equality:
        - contributionPhase() == false && belowFloorAsked() == true
(LiquidHelper.sol#50)
LiquidLocker.liquidateLocker() (LiquidLocker.sol#487-511) uses a dangerous strict
equality:
        - require(bool,string)(missedActivate() == true || missedDeadline() ==
true,LiquidLocker: TOO_EARLY) (LiquidLocker.sol#490-494)
LiquidLocker.onlyDuringContributionPhase() (LiquidLocker.sol#26-33) uses a dangerous
strict equality:
        - require(bool,string)(contributionPhase() == true && paymentTimeNotSet() ==
true,LiquidLocker: INVALID_PHASE) (LiquidLocker.sol#27-31)
LiquidLocker.payBackFunds(uint256,address) (LiquidLocker.sol#429-481) uses a dangerous
strict equality:
        - require(bool,string)(missedDeadline() == false,LiquidLocker: TOO_LATE)
(LiquidLocker.sol#436-439)
LiquidLocker.payBackFunds(uint256,address) (LiquidLocker.sol#429-481) uses a dangerous
strict equality:
        - payedTimestamp == finalTimestamp (LiquidLocker.sol#462)
LiquidHelper.paymentTimeNotSet() (LiquidHelper.sol#127-133) uses a dangerous strict
equality:
        - nextDueTime == 0 (LiquidHelper.sol#132)
LiquidLocker.refundDueExpired(address) (LiquidLocker.sol#367-388) uses a dangerous
strict equality:
        - require(bool,string)(floorNotReached() == true || ownerlessLocker() ==
true,LiquidLocker: ENABLED_LOCKER) (LiquidLocker.sol#372-376)
LiquidLocker.rescueLocker() (LiquidLocker.sol#343-362) uses a dangerous strict
equality:
        - require(bool,string)(paymentTimeNotSet() == true,LiquidLocker:
ALREADY_STARTED) (LiquidLocker.sol#356-359)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities
```

```
Reentrancy in LiquidLocker.refundDueExpired(address) (LiquidLocker.sol#367-388):
        External calls:
        - _refundTokens(tokenAmount,_refundAddress) (LiquidLocker.sol#380-383)
                - (success,data) =
_token.call(abi.encodeWithSelector(TRANSFER,_to,_value)) (LiquidHelper.sol#282-288)
        State variables written after the call(s):
        - _decreaseTotalCollected(tokenAmount) (LiquidLocker.sol#385-387)
                - totalCollected = totalCollected - _decreaseAmount
(LiquidHelper.sol#247-248)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1

LiquidLocker.initialize(uint256[],address,address,uint256,uint256,uint256,uint256)
(LiquidLocker.sol#39-62) should emit an event for:
        - totalAsked = _totalAsked (LiquidLocker.sol#60)
LiquidLocker.donateFunds(uint256) (LiquidLocker.sol#413-421) should emit an event for:
        - claimableBalance = claimableBalance + _donationAmount
(LiquidLocker.sol#419-420)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

LiquidTransfer._transferNFT(address,address,address,uint256) (LiquidTransfer.sol#20-59)
has external calls inside a loop: (success) = address(_tokenAddress).call(data)
(LiquidTransfer.sol#51-53)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop

Reentrancy in LiquidLocker._claimInterest(address) (LiquidLocker.sol#661-685):
        External calls:
        - _safeTransfer(PAYMENT_TOKEN,_claimAddress,tokensToTransfer)
(LiquidLocker.sol#675-679)
                - (success,data) =
_token.call(abi.encodeWithSelector(TRANSFER,_to,_value)) (LiquidHelper.sol#282-288)
        Event emitted after the call(s):
        - ClaimMade(tokensToTransfer,_claimAddress) (LiquidLocker.sol#681-684)
Reentrancy in LiquidLocker._refundTokens(uint256,address) (LiquidLocker.sol#690-709):
        External calls:
        - _safeTransfer(PAYMENT_TOKEN,_refundAddress,_refundAmount)
(LiquidLocker.sol#699-703)
                - (success,data) =
_token.call(abi.encodeWithSelector(TRANSFER,_to,_value)) (LiquidHelper.sol#282-288)
```

```
        Event emitted after the call(s):
        - RefundMade(_refundAmount,_refundAddress) (LiquidLocker.sol#705-708)
Reentrancy in LiquidLocker.enableLocker(uint256) (LiquidLocker.sol#245-299):
        External calls:
        - _safeTransfer(PAYMENT_TOKEN,msg.sender,totalCollected - _prepayAmount -
teamsPayback) (LiquidLocker.sol#283-287)
                - (success,data) =
_token.call(abi.encodeWithSelector(TRANSFER,_to,_value)) (LiquidHelper.sol#282-288)
        - _safeTransfer(PAYMENT_TOKEN,TRUSTEE_MULTISIG,teamsPayback)
(LiquidLocker.sol#289-293)
                - (success,data) =
_token.call(abi.encodeWithSelector(TRANSFER,_to,_value)) (LiquidHelper.sol#282-288)
        Event emitted after the call(s):
        - PaymentMade(_prepayAmount,msg.sender) (LiquidLocker.sol#295-298)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

LiquidHelper.floorNotReached() (LiquidHelper.sol#45-51) uses timestamp for comparisons
        Dangerous comparisons:
        - contributionPhase() == false && belowFloorAsked() == true
(LiquidHelper.sol#50)
LiquidHelper.missedActivate() (LiquidHelper.sol#86-94) uses timestamp for comparisons
        Dangerous comparisons:
        - floorNotReached() && startingTimestamp() + DEADLINE_TIME < block.timestamp
(LiquidHelper.sol#91-93)
LiquidHelper.missedDeadline() (LiquidHelper.sol#99-111) uses timestamp for comparisons
        Dangerous comparisons:
        - nextDueTime > 0 && nextDueOrDeadline + DEADLINE_TIME < block.timestamp
(LiquidHelper.sol#108-110)
LiquidHelper.contributionPhase() (LiquidHelper.sol#138-144) uses timestamp for
comparisons
        Dangerous comparisons:
        - timeSince(creationTime) < CONTRIBUTION_TIME (LiquidHelper.sol#143)
LiquidLocker.enableLocker(uint256) (LiquidLocker.sol#245-299) uses timestamp for
comparisons
        Dangerous comparisons:
        - require(bool,string)(paymentTimeNotSet() == true,LiquidLocker:
ENABLED_LOCKER) (LiquidLocker.sol#256-259)
LiquidLocker.rescueLocker() (LiquidLocker.sol#343-362) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(timeSince(creationTime) > DEADLINE_TIME,LiquidLocker:
```

```
NOT_ENOUGHT_TIME) (LiquidLocker.sol#351-354)
        - require(bool,string)(paymentTimeNotSet() == true,LiquidLocker:
ALREADY_STARTED) (LiquidLocker.sol#356-359)
LiquidLocker.refundDueExpired(address) (LiquidLocker.sol#367-388) uses timestamp for
comparisons
        Dangerous comparisons:
        - require(bool,string)(floorNotReached() == true || ownerlessLocker() ==
true,LiquidLocker: ENABLED_LOCKER) (LiquidLocker.sol#372-376)
LiquidLocker.payBackFunds(uint256,address) (LiquidLocker.sol#429-481) uses timestamp
for comparisons
        Dangerous comparisons:
        - require(bool,string)(missedDeadline() == false,LiquidLocker: TOO_LATE)
(LiquidLocker.sol#436-439)
        - payedTimestamp == finalTimestamp (LiquidLocker.sol#462)
        - payedTimestamp > block.timestamp (LiquidLocker.sol#476-478)
LiquidLocker.liquidateLocker() (LiquidLocker.sol#487-511) uses timestamp for
comparisons
        Dangerous comparisons:
        - require(bool,string)(missedActivate() == true || missedDeadline() ==
true,LiquidLocker: TOO_EARLY) (LiquidLocker.sol#490-494)
LiquidLocker.getLateDays() (LiquidLocker.sol#580-587) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > nextDueTime (LiquidLocker.sol#585-586)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

LiquidHelper.floorNotReached() (LiquidHelper.sol#45-51) compares to a boolean constant:
        -contributionPhase() == false && belowFloorAsked() == true
(LiquidHelper.sol#50)
LiquidLocker.enableLocker(uint256) (LiquidLocker.sol#245-299) compares to a boolean
constant:
        -require(bool,string)(belowFloorAsked() == false,LiquidLocker: BELOW_FLOOR)
(LiquidLocker.sol#251-254)
LiquidLocker.enableLocker(uint256) (LiquidLocker.sol#245-299) compares to a boolean
constant:
        -require(bool,string)(paymentTimeNotSet() == true,LiquidLocker: ENABLED_LOCKER)
(LiquidLocker.sol#256-259)
LiquidLocker.disableLocker() (LiquidLocker.sol#305-315) compares to a boolean constant:
        -require(bool,string)(belowFloorAsked() == true,LiquidLocker: FLOOR_REACHED)
(LiquidLocker.sol#309-312)
LiquidLocker.rescueLocker() (LiquidLocker.sol#343-362) compares to a boolean constant:
```

```
        -require(bool,string)(paymentTimeNotSet() == true,LiquidLocker:
ALREADY_STARTED) (LiquidLocker.sol#356-359)
LiquidLocker.refundDueExpired(address) (LiquidLocker.sol#367-388) compares to a boolean
constant:
        -require(bool,string)(floorNotReached() == true || ownerlessLocker() ==
true,LiquidLocker: ENABLED_LOCKER) (LiquidLocker.sol#372-376)
LiquidLocker.refundDueSingle(address) (LiquidLocker.sol#393-407) compares to a boolean
constant:
        -require(bool,string)(notSingleProvider(_refundAddress) == true,LiquidLocker:
INVALID_SENDER) (LiquidLocker.sol#398-401)
LiquidLocker.payBackFunds(uint256,address) (LiquidLocker.sol#429-481) compares to a
boolean constant:
        -require(bool,string)(missedDeadline() == false,LiquidLocker: TOO_LATE)
(LiquidLocker.sol#436-439)
LiquidLocker.liquidateLocker() (LiquidLocker.sol#487-511) compares to a boolean
constant:
        -require(bool,string)(missedActivate() == true || missedDeadline() ==
true,LiquidLocker: TOO_EARLY) (LiquidLocker.sol#490-494)
LiquidLocker.onlyDuringContributionPhase() (LiquidLocker.sol#26-33) compares to a
boolean constant:
        -require(bool,string)(contributionPhase() == true && paymentTimeNotSet() ==
true,LiquidLocker: INVALID_PHASE) (LiquidLocker.sol#27-31)
LiquidTransfer._transferNFT(address,address,address,uint256) (LiquidTransfer.sol#20-59)
compares to a boolean constant:
        -require(bool,string)(success == true,NFT_TRANSFER_FAILED)
(LiquidTransfer.sol#55-58)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-
equality

LiquidHelper._safeTransferFrom(address,address,address,uint256)
(LiquidHelper.sol#303-328) is never used and should be removed
LiquidTransfer._transferFromNFT(address,address,address,uint256)
(LiquidTransfer.sol#64-139) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version=0.8.12 (LiquidBase.sol#3) necessitates a version too recent to be
trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version=0.8.12 (LiquidHelper.sol#3) necessitates a version too recent to be
trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version=0.8.12 (LiquidLocker.sol#3) necessitates a version too recent to be
trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
```

```
Pragma version=0.8.12 (LiquidTransfer.sol#3) necessitates a version too recent to be
trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Low level call in LiquidHelper._safeTransfer(address,address,uint256)
(LiquidHelper.sol#275-298):
        - (success,data) = _token.call(abi.encodeWithSelector(TRANSFER,_to,_value))
(LiquidHelper.sol#282-288)
Low level call in LiquidHelper._safeTransferFrom(address,address,address,uint256)
(LiquidHelper.sol#303-328):
        - (success,data) =
_token.call(abi.encodeWithSelector(TRANSFER_FROM,_from,_to,_value))
(LiquidHelper.sol#311-318)
Low level call in LiquidTransfer._transferNFT(address,address,address,uint256)
(LiquidTransfer.sol#20-59):
        - (success) = address(_tokenAddress).call(data) (LiquidTransfer.sol#51-53)
Low level call in LiquidTransfer._transferFromNFT(address,address,address,uint256)
(LiquidTransfer.sol#64-139):
        - (checkSuccess,result) = address(_tokenAddress).staticcall(punkIndexToAddress)
(LiquidTransfer.sol#87-89)
        - (buySuccess,buyResultData) = address(_tokenAddress).call(buyData)
(LiquidTransfer.sol#107-109)
        - (success,resultData) = address(_tokenAddress).call(data)
(LiquidTransfer.sol#131-133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Parameter LiquidHelper.notSingleProvider(address)._checkAddress (LiquidHelper.sol#57)
is not in mixedCase
Parameter LiquidHelper.reachedTotal(address,uint256)._contributor (LiquidHelper.sol#73)
is not in mixedCase
Parameter LiquidHelper.reachedTotal(address,uint256)._tokenAmount (LiquidHelper.sol#74)
is not in mixedCase
Parameter LiquidHelper.timeSince(uint256)._timeStamp (LiquidHelper.sol#196) is not in
mixedCase
Parameter LiquidLocker.initialize(uint256[],address,address,uint256,uint256,uint256,uint
256)._tokenId (LiquidLocker.sol#40) is not in mixedCase
Parameter LiquidLocker.initialize(uint256[],address,address,uint256,uint256,uint256,uint
256)._tokenAddress (LiquidLocker.sol#41) is not in mixedCase
```

```
Parameter LiquidLocker.initialize(uint256[],address,address,uint256,uint256,uint256,uint
256)._tokenOwner (LiquidLocker.sol#42) is not in mixedCase
Parameter LiquidLocker.initialize(uint256[],address,address,uint256,uint256,uint256,uint
256)._floorAsked (LiquidLocker.sol#43) is not in mixedCase
Parameter LiquidLocker.initialize(uint256[],address,address,uint256,uint256,uint256,uint
256)._totalAsked (LiquidLocker.sol#44) is not in mixedCase
Parameter LiquidLocker.initialize(uint256[],address,address,uint256,uint256,uint256,uint
256)._paymentTime (LiquidLocker.sol#45) is not in mixedCase
Parameter LiquidLocker.initialize(uint256[],address,address,uint256,uint256,uint256,uint
256)._paymentRate (LiquidLocker.sol#46) is not in mixedCase
Parameter LiquidLocker.increasePaymentRate(uint256)._newPaymntRate
(LiquidLocker.sol#68) is not in mixedCase
Parameter LiquidLocker.decreasePaymentTime(uint256)._newPaymentTime
(LiquidLocker.sol#92) is not in mixedCase
Parameter LiquidLocker.updateSettings(uint256,uint256)._newPaymntRate
(LiquidLocker.sol#114) is not in mixedCase
Parameter LiquidLocker.updateSettings(uint256,uint256)._newPaymentTime
(LiquidLocker.sol#115) is not in mixedCase
Parameter LiquidLocker.makeContribution(uint256,address)._tokenAmount
(LiquidLocker.sol#151) is not in mixedCase
Parameter LiquidLocker.makeContribution(uint256,address)._tokenHolder
(LiquidLocker.sol#152) is not in mixedCase
Parameter LiquidLocker.enableLocker(uint256)._prepayAmount (LiquidLocker.sol#246) is
not in mixedCase
Parameter LiquidLocker.refundDueExpired(address)._refundAddress (LiquidLocker.sol#368)
is not in mixedCase
Parameter LiquidLocker.refundDueSingle(address)._refundAddress (LiquidLocker.sol#394)
is not in mixedCase
Parameter LiquidLocker.donateFunds(uint256)._donationAmount (LiquidLocker.sol#414) is
not in mixedCase
Parameter LiquidLocker.payBackFunds(uint256,address)._paymentAmount
(LiquidLocker.sol#430) is not in mixedCase
Parameter LiquidLocker.payBackFunds(uint256,address)._paymentAddress
(LiquidLocker.sol#431) is not in mixedCase
Parameter LiquidLocker.penaltyAmount(uint256,uint256)._totalCollected
(LiquidLocker.sol#517) is not in mixedCase
Parameter LiquidLocker.penaltyAmount(uint256,uint256)._lateDaysAmount
(LiquidLocker.sol#518) is not in mixedCase
Parameter LiquidLocker.calculatePaybacks(uint256,uint256,uint256)._totalValue
(LiquidLocker.sol#595) is not in mixedCase
Parameter LiquidLocker.calculatePaybacks(uint256,uint256,uint256)._paymentTime
```

```
(LiquidLocker.sol#596) is not in mixedCase
Parameter LiquidLocker.calculatePaybacks(uint256,uint256,uint256)._paymentRate
(LiquidLocker.sol#597) is not in mixedCase
Parameter LiquidLocker.calculateEpoch(uint256,uint256,uint256)._totalValue
(LiquidLocker.sol#623) is not in mixedCase
Parameter LiquidLocker.calculateEpoch(uint256,uint256,uint256)._paymentTime
(LiquidLocker.sol#624) is not in mixedCase
Parameter LiquidLocker.calculateEpoch(uint256,uint256,uint256)._paymentRate
(LiquidLocker.sol#625) is not in mixedCase
Parameter LiquidTransfer.onERC721Received(address,address,uint256,bytes)._operator
(LiquidTransfer.sol#149) is not in mixedCase
Parameter LiquidTransfer.onERC721Received(address,address,uint256,bytes)._from
(LiquidTransfer.sol#150) is not in mixedCase
Parameter LiquidTransfer.onERC721Received(address,address,uint256,bytes)._tokenId
(LiquidTransfer.sol#151) is not in mixedCase
Parameter LiquidTransfer.onERC721Received(address,address,uint256,bytes)._data
(LiquidTransfer.sol#152) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions

getTokens() should be declared external:
        - LiquidHelper.getTokens() (LiquidHelper.sol#34-40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
LiquidLocker.sol analyzed (4 contracts with 77 detectors), 84 result(s) found
```

0x Guard